IN THE SPEC IFICATION

Please amend the paragraph beginning on page 1, line 5 as follows:

--There are three significant solutions known to the inventor in the prior art that relate to ~~improve~~ improving SMI latency on systems using timing-sensitive registers. The first is to ensure that the update-in-progress status bit is clear before exiting an SMI handler.--

Please amend the paragraph beginning on page 1, line 15 as follows:

--Disadvantages of prior art are as follows. The first prior art solution requires that in 0.1% of SMIs, the additional latency will be up to 1 ms. The second prior art solution requires that all code adhere to the standard. Unfortunately, quite a bit of legacy code accesses the real-time clock registers without any regard to this standard. The third prior art solution contains a race condition where, between the reading of the system-wide flag and the access of the data register, an SMI occurs. ~~This~~ In such a situation, the code would proceed to access the possibly-corrupted register.--

Please delete the paragraph beginning on page 2, line 1 in its entirety.

Please amend the paragraph beginning on page 2, line 6 as follows:

~~--To meet the above and other objectives, the present invention comprises systems, methods, and software that improve SMI latency on systems that use timing-sensitive registers.~~ The present invention ~~comprising~~ includes a real-time clock, a register file containing one or more timing-sensitive registers, and an index and data register for accessing the timing sensitive registers in the register file. An update-in-progress status bit determines a certain fixed period of time for which the timing-sensitive registers are valid. --

Please amend the paragraph beginning on page 3, line 20 as follows:

--The system 10 includes a retriggerable, fixed duration timer 15 that is triggered by reads of zero from the update-in-progress status bit 14. The retriggerable, fixed duration timer 15 has a duration that is fixed at ~1 ms. This duration depends on the length of the no-update time (~244 μs, for example) plus the actual update time. If the retriggerable, fixed duration timer 15 expires, it does not restart. The status of the retriggerable, fixed duration timer 15 ~~timer 15~~ running or not, can be determined. The status of the retriggerable, fixed duration timer 15 may be determined by reading a status register that is contained in a memory-mapped I/O location, I/O register, or PCI configuration register.~~.~~ --

Please amend the paragraph beginning on page 4, line 16 as follows:

--The system management interrupt (SMI) is the highest priority interrupt in some families of x86 central processing units (CPUs). It cannot be disabled using normal CPU interrupt-prevention mechanisms. There is a boundary condition where an SMI occurs after step 3. Depending on the length of time requires to service the SMI, steps 4 and 5 might not be executed until after the 244 ~~μ4s~~ <u>μs</u> window of safety has passed, this causing potential data corruption. --

Please amend the paragraph beginning on page 4, line 33 as follows:

--The present invention relies<u>, at least in part, on</u> the observation that the period of potential danger occurs for ~1 ms after the read of the update-in-progress status bit 14 with a value of zero. The present invention adds a timer 20 which is triggered by reading zero from register location 0xA, bit 7. The status of the timer 20 (running or not) is stored in a status latch 21 upon assertion of SMI and the latched value can be read by software code (such as the SMI handler 18) using a status bit 22. This status bit 22 is the indicator of whether the SMI occurred during this period of potential danger. If the SMI handler 18 was in the critical period, then it must check the update-in-progress bit 14 before proceeding. Otherwise, the SMI handler 18 may exit normally. The status latch 21 is cleared when the SMI is deasserted. --

Please amend the paragraph beginning on page 5, line 6 as follows:

--The steps of an exemplary method 30 and that is also implemented in software (SMI handler 18) in accordance with the principles of the present invention are as follows.

Step 1 is to read 31 the status latch 21.

In step 2, if the status latch 21 is zero, stop 32 (exit ~~32~~ the SMI handler 18).

Step ~~3~~ 3 ~~s to~~, if the value of the status latch 21 is non-zero, write 33 0A to I/O location 0x70.

Step 4 is to read 34 I/O location 0x71.

In step 5, if bit 7 of the value read is set, go 35 (jump ~~35~~) to step 3.

Step 6 is to step ~~32~~ 36 (exit ~~32~~ the SMI handler 18).

Since the sequence of reading the update-in-progress status bit 14 is rate, the worst-case SMI latency is the same, but the average SMI latency is improved. --